Name: Pearl Law
SCU ID: W0958839
Rank: 4
MCC-score: 0.8748

Peptide Classification

Abstract

The purpose of this project is to explore and develop predictive neural networks for binary peptide classification. Two multi-layer perceptron models were implemented with Python and Tensorflow separately to classify biofilm and antibiofilm peptide sequences. Both models utilized the Adam optimizer along with several fine-tuned model hyperparameters (i.e., learning rate, regularization, etc.) to optimize the model's performance. The trained Python model yielded ~99.9% accuracy on the training dataset and an MCC score = 0.8748, indicating that the trained model performs quite accurately in the binary classification task. The trained Tensorflow model performed similarly, with ~100% accuracy on the training dataset.

Introduction

Proteins carry out many important functions within the body such as driving metabolic reactions, DNA replication, and transporting molecules throughout the body. Biofilms are one of the major causes of disease due to their robust multicellular matrices, which make them resistant to host and antibiotic defenses. Thus, it is important to understand which proteins are antibiofilm. The goal of this project is to explore and develop predictive neural networks to classify whether a particular antibacterial peptide is also a biofilm peptide. There are 1566 peptide samples in the training dataset, each with a corresponding antibiofilm (1) or not antibiofilm (-1) label. There are 392 peptide sequences in the test dataset to be classified. This explorative study examines the effect of different model parameters and optimization functions to achieve accurate classification of biofilm peptides.

Approach

k-mers was implemented to identify a total of 7090 three-sequence amino acid features within each peptide sequence in the training and test datasets. All instances of X amino acid were replaced with V in the training dataset. Oversampling was applied to balance the positive and negative class samples in the original unbalanced training dataset. This procedure increased the training dataset to 2848 samples (1424 samples from each class). The training and test datasets were standardized using MinMaxScaler from the scikit-learn library. The datasets were reshaped, and the negative class labels were encoded from -1 to 0 before training the model on the dataset. The class labels were decoded from 0 to -1 after classification.

The first model is a two-layer neural network implemented using Python and Numpy library. The model applies the Adam optimizer with parameters beta1 = 0.9, beta2 = 0.999, and epsilon

= 1e-8. The network consists of a first layer using ReLU activation and output size 100 and a second layer using Sigmoid activation and output size 1. At the start, the model weights are He initialized, and biases are zero initialized. Adam variables are also zero initialized at the start to keep track of weighted average of past gradients and squares of past gradients. Regularization (with lambda = 1e-3) is added to the binary cross-entropy loss function and applied to backward propagation so that gradients are computed with respect to this new loss. The model is trained for 10000 iterations at a learning rate of 3e-4.

The second model is a two-layer neural network implemented using Tensorflow. The first layer contains weights that are He initialized and uses ReLU activation function with output size 30. L2 regularization is applied to the weights with lambda = 1e-3. 10% of neurons are dropped out after the first layer. The last layer uses Sigmoid function with output size 1 because the goal is to perform binary classification. The model is configured with Adam optimizer function and binary cross-entropy loss function. Early stopping is implemented to stop training while maintaining maximum accuracy when there is no further improvement in validation loss for 10 consecutive epochs. The model is trained with the training dataset for 100 epochs or until early stopping at a learning rate of 1e-3. 20% of the total training dataset is held out for validation.

Methodology

The model weights are He initialized because compared to zero initialization which produces zeroed/dead neurons and random initialization which can slow down optimization if random numbers are too large, He initialization considers the non-linearity of activation functions such as ReLU, which is what is used in this model. L2 regularization is introduced to ensure the model is not overfitting the training dataset. Lambda values ranging from 1e-1 to 1e-4 were tested and lambda = 1e-3 was chosen because it generalized the best to the test data. Various optimization methods were tested including gradient descent (with and without momentum), mini batch gradient descent, and Adam. Gradient descent and mini batch gradient descent performed similarly well in. Gradient descent with momentum performed slightly better than gradient descent without momentum primarily since momentum considers past gradients to smoothen out gradient descent. However, of all the optimization methods attempted, Adam performed significantly better (highest accuracy) than the rest because it combines RMSProp with momentum to handle the sparse feature matrix present in this dataset. An optimal learning rate of 3e-4 was observed after tuning the learning rate over the range 1e-3 to 1e-5.

Conclusion

The trained Python model yielded ~99.9% accuracy on the training dataset and an MCC score = 0.8748, indicating that the trained model performs quite accurately in the binary classification task. The trained Tensorflow model performed similarly, with ~100% accuracy on the training dataset. The results from this project demonstrate that Adam optimization is critical towards optimizing models trained on sparse datasets. A two-layer neural network that combines Adam with a learning rate of 3e-4 and various other hyperparameter optimizations produces well-performing classifiers.